Kalman Filter for attitude estimation
DroMOOC data set

The *attitude_estimation* ROS package provides different data set recorded from an Inertial Measurement Unit (IMU) and implements a Kalman Filter (KF) for estimation of attitude angles, as well as graphical tools for parameter tuning and visualization.

Detailed instructions on how to install and run the Virtual Machine can be found in the document "Importing and starting the DroMOOC Virtual Machine".

The *attitude_estimation* package is located in the `~/catkin_ws/src/attitude_estimation` directory of the Virtual Machine and in the DroMOOC Github repository. Before first use, it is recommended to update the package located on your Virtual Machine by entering the following command lines in a terminal (internet connection is required):

```
roscd attitude_estimation
git pull
```

If you need some help, use the contact us link on the website.

1. Understanding the data set

Measurements provided by an Inertial Measurement Unit (IMU) have been recorded in the following conditions to provided different data sets:

- still horizontal IMU (no motion): no_motion.bag
- small rotations around pitch axis[1]: pitch_only.bag
- small rotations around roll, pitch and yaw axis: roll_pitch_yaw.bag

Data sets have been recorded and can be played using the rosbag tools. They are located in the directory *bags* of the *attitude_estimation* package.

Measurements are recorded as Imu and MagneticField messages defined by the ROS sensor_msgs package and hence contain the following time-stamped information:

- 3 axis accelerometer measurements (**linear accelerations**)
- 3 axis rate-gyro measurements (**angular velocities**)
- 3 axis magnetometer measurements (**magnetic field**)
- Quaternion computed by the internal processing of the IMU

The **attitude estimation problem** consists in computing **roll, pitch and yaw angles** (see Figure 1) from the sensor measurements. Estimated angles will be compared to angles deduced from the quaternion computed by the internal processing of the IMU, which will be assumed accurate enough to be considered as ground truth.

[1] Data sets are also provided for small rotations around roll axis (roll_only.bag) and yaw axis (yaw_only.bag) only, and for rotations around the three axes (roll_pitch_yaw.bag).

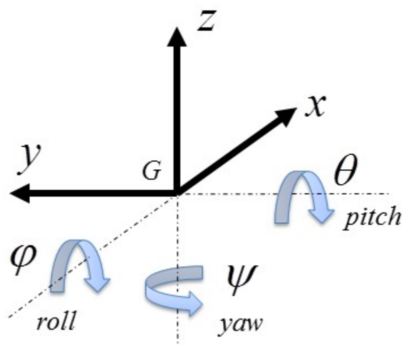


Figure 1. Simplified definition of Roll-Pitch-Yaw

2. Characteristics of the sensors

The first data set (*no_motion.bag*) can be used to analyze some characteristics of the sensors, namely their bias and noise variance.

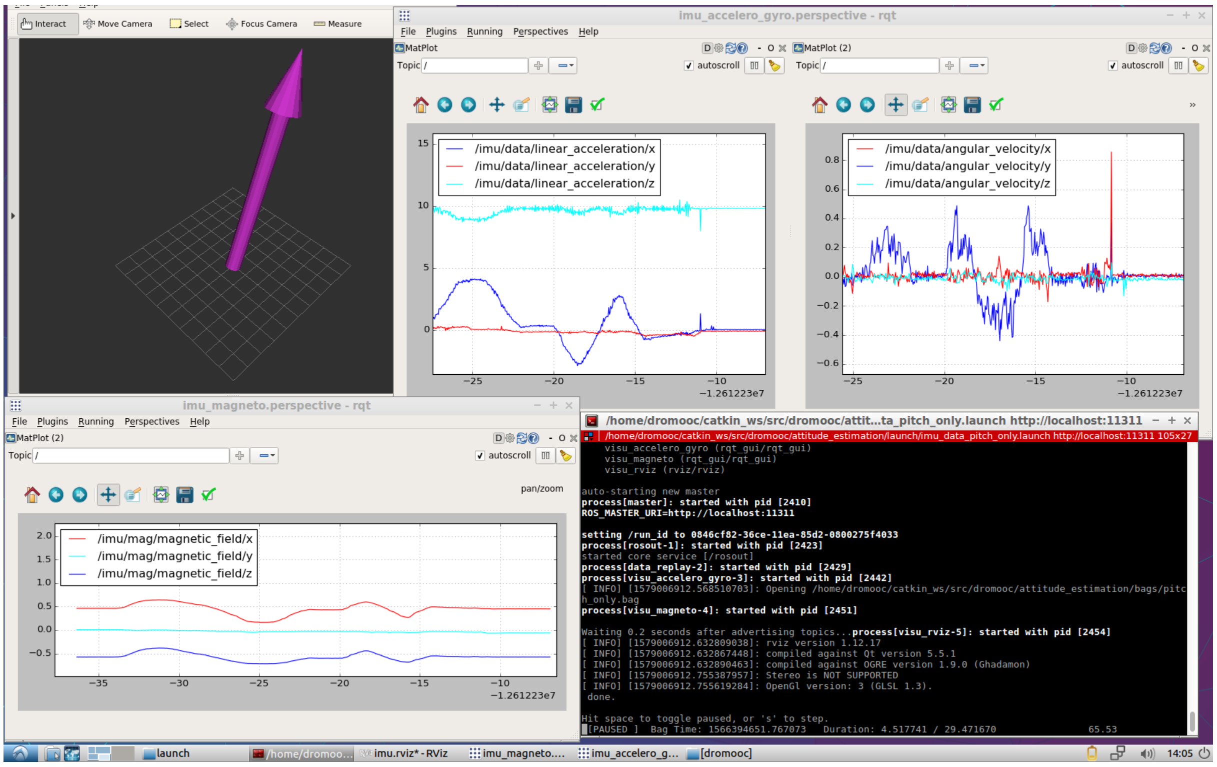
To play this record and visualize the data, open a new terminal and enter the command line:

```
roslaunch attitude_estimation imu_data_no_motion.launch
```

It will open two graphical interfaces with time plots of the acceleration, angular velocity and magnetic field measurements (see Figure 2). A RViz window also presents a 3D arrow which corresponds to the normal (local z-axis) of the IMU. This arrow will remain aligned to the "vertical" for the data set with no motion.

The data recorded in this bag are also made available in a CSV file for more convenience.

You can use this CSV file directly to compute the mean and variance of the acceleration and angular velocity measurements over time. It will provide information on the bias of the sensors and variance of their noise. Note that acceleration due to gravity must be substracted from the acceleration signal along the z-axis.


imu_accelero_gyro.perspective - rqt
/imu/data/linear_acceleration/x
/imu/data/linear_acceleration/y
/imu/data/linear_acceleration/z
/imu/data/angular_velocity/x
/imu/data/angular_velocity/y
/imu/data/angular_velocity/z
imu_magneto.perspective - rqt
/imu/mag/magnetic_field/x
/imu/mag/magnetic_field/y
/imu/mag/magnetic_field/z
−1.261223e7

```
/home/dromooc/catkin_ws/src/dromooc/attitude_estimation/launch/imu_data_pitch_only.launch http://localhost:11311 105x27
    visu_accelero_gyro (rqt_gui/rqt_gui)
    visu_magneto (rqt_gui/rqt_gui)
    visu_rviz (rviz/rviz)

auto-starting new master
process[master]: started with pid [2410]
ROS_MASTER_URI=http://localhost:11311

setting /run_id to 0846cf82-36ce-11ea-85d2-0800275f4033
process[rosout-1]: started with pid [2423]
started core service [/rosout]
process[data_replay-2]: started with pid [2429]
process[visu_accelero_gyro-3]: started with pid [2442]
[ INFO] [1579006912.568510703]: Opening /home/dromooc/catkin_ws/src/dromooc/attitude_estimation/bags/pitc
h_only.bag
process[visu_magneto-4]: started with pid [2451]

Waiting 0.2 seconds after advertising topics...process[visu_rviz-5]: started with pid [2454]
[ INFO] [1579006912.632809038]: rviz version 1.12.17
[ INFO] [1579006912.632867448]: compiled against Qt version 5.5.1
[ INFO] [1579006912.632890463]: compiled against OGRE version 1.9.0 (Ghadamon)
[ INFO] [1579006912.755387957]: Stereo is NOT SUPPORTED
[ INFO] [1579006912.755619284]: OpenGl version: 3 (GLSL 1.3).
 done.

Hit space to toggle paused, or 's' to step.
 [PAUSED ]  Bag Time: 1566394651.767073   Duration: 4.517741 / 29.471670          65.53
```

Figure 2. Graphical interfaces for visualization of data sets
(example on the *pitch_only.bag* data set)

3. Pitch estimation using Kalman Filter

3.1. Data set

You can then visualize the second data set related to pitch motion by opening a new terminal and entering the command line:

```
roslaunch attitude_estimation imu_data_pitch_only.launch
```

3.2. Python implementation of a Kalman Filter

As presented in the video lectures, a Kalman Filter will be used to estimate online the pitch angle from the measurements of the data set. A Python implementation of a linear Kalman Filter is provided in the script *KalmanFilter.py* located in the *script* directory of the *attitude_estimation* package. Take some time to understand this implementation and the way it can be used by looking at the example at the end of this script. You can run the script to visualize some graphical results on this example.

Note that this implementation is generic in the sense that you can use it for other purposes by simply importing the KalmaFilter module in any other Python code (*import KalmanFilter*).

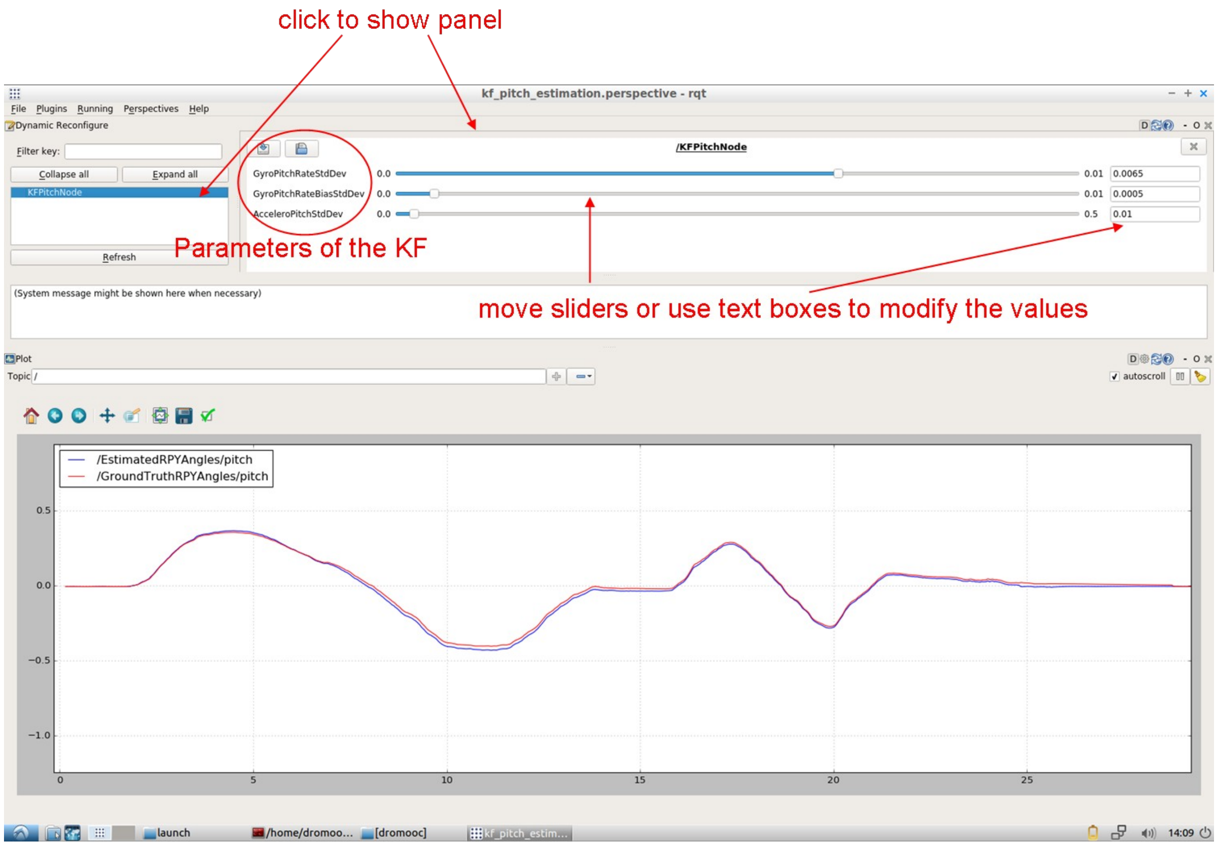
3.3. Use of the Kalman Filter for pitch estimation

The Python file KFPitchEstimation.py implements a ROS node with the Kalman Filter for pitch estimation. You can try the code by opening a new terminal and entering the command line:

```
roslaunch attitude_estimation KF_pitch_imu_data_pitch_only.launch
```

It will play the pitch_only.bag data set, run the node implemented in KFPitchEstimation.py and open a graphical user interface that allows online tuning of the parameters

This command line starts the replay of the *pitch_only.bag* data set, runs the node implemented in *KFPitchEstimation.py* and also starts a graphical user interface based on RQT (Figure 3). This interface enables to tune parameters of the filter (Figure 3 – top), and to plot the time evolutions of the "true" pitch angle to be estimated (ground truth – in red) and of the pitch angle estimated by the Kalman filter (in blue).



GyroPitchRateStdDev	0.0	0.01	0.0065
GyroPitchRateBiasStdDev	0.0	0.01	0.0005
AcceleroPitchStdDev	0.0	0.5	0.01

(System message might be shown here when necessary)

Figure 3 – Graphical user interface associated to the Kalman Filter for pitch estimation

3.4. Understanding the code

The simulator package is composed of several directories:

- `scripts`: Python files for Kalman filter and pitch estimation node
- `bags`: bags file with data sets

- `msg`: custom message formats used by the code
- `launch`: launch files that are used to run several nodes
- `doc`: this document

The node for KF pitch estimation (/KFPitchNode) requires acceleration and angular velocity measurements provided by the IMU. They are given by the /imu/data topic generated by the replay of the bag file (/data_replay node).

The node for KF pitch estimation publishes the estimated pitch angle (/EstimatedRPYAngles/ pitch topic) and is also responsible for the publication of the ground truth value of the pitch angle (/GroundTruthRPYAngles/pitch topic) computed from the quaternion provided by the internal processing of the IMU.

The main structure of the code is summarized by the following Node Graph automatically generated by ROS:

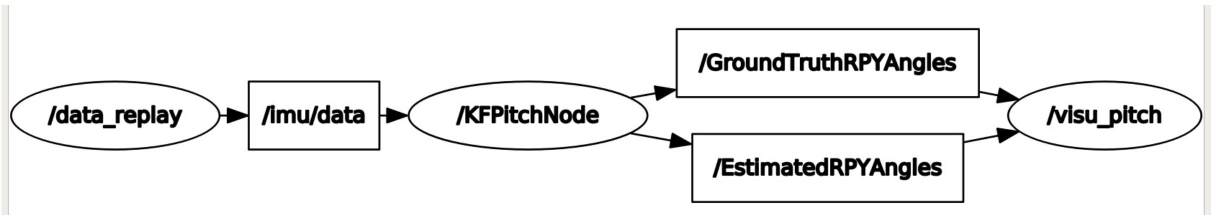


Figure 4 – Node Graph of the pitch estimation implementation

4. Proposed work (“homework”)

The *attitude_estimation* package is provided as standalone additional material to DroMOOC to understand the basis of Kalman filtering and its application to attitude angle estimation, with illustration on pitch angle.

Three simple and direct extensions are proposed below as “homework”.

Estimation of the roll angle

Develop a new node for roll angle estimation (*KFPitchEstimation.py*) using a Kalman filter. Take the code for pitch estimation as a basis and implement the equations given in the video lectures for the roll angle. Test your code by using the data set recorded in the *roll_only.bag* file.

Estimation of the yaw angle

Develop a new node for yaw angle estimation (YawEstimation.py) using the geometrical equations given in the video lectures. Test your code by using the data set recorded in the *yaw_only.bag* file.

Estimation of the roll pitch and yaw angles

Test the code provided for pitch estimation and the codes you have developed for roll and yaw estimation by running simultaneously these three nodes and using the data set recorded in the *roll_pitch_yaw.bag* file.

If you need some help, use the contact us link on the website.